6.897: CSCI8980 Algorithmic Techniques for Big Data October 4, 2013

Lecture 8

Dr. Barna Saha *Scribe: Mohit Sharma*

Overview

In this lecture we will look at the methods of clustering. Here we will discuss K-Center and K-Median algorithms for clustering data stream and K-Means++ algorithm which provides improved seeding of conventional K-Means algorithm.

1 Introduction

Consider a set of distinct points $P = \{p_1, p_2, ..., p_n\}$. Our goal is to find a set of k points $Q \subset P$, $|Q| = k$ that minimizes a given objective function. We will call these k points as cluster centers. All remaining points in P will belong to a cluster centered at one of these k points.

2 K-Center

Points from set $P = \{p_1, p_2, ..., p_n\}$ are coming in streaming fashion. We will find a set of k points $Q \subset P$, $|Q| = k$, that minimizes

$$\max_i \min_{q \in Q} d(p_i, q)$$

where d is any distance metric. Above objective function minimizes the maximum cluster radius or minimizes the maximum distance travelled by a point to the center of assigned cluster. Suppose the above optimal distance is r. If we know r, then we can find 2-approx solution to the above problem in $O(k)$ space.

2.1 Thresholded Algorithm

This algorithm gives a 2-approx solution provided we know r. When a new point from stream comes then we compute the minimum distance of this point from already opened centers. If this minimum distance is more than $2r$ then we will open a new center at that point, otherwise assigned this point to the nearest open center. If we don't have any opened center then we use the coming point as our new center.

Claim 1. *We will open at most k centers, if we know r exactly and k is given.*

Proof. Suppose we are processing the stream and we have opened k centers so far i.e. $Q = \{q_1, q_2, ..., q_k\}$. Say a new point p_j comes from the stream and its distance from all the k centers is greater than 2r. Then we will open a new center at p_j and we will now have k+1 centers.

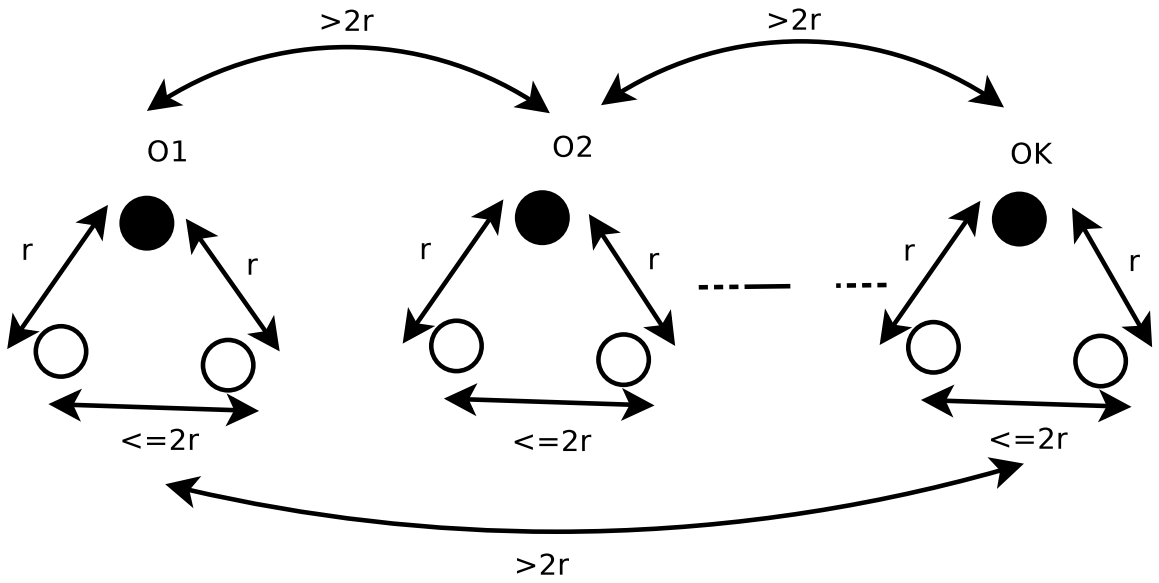

Figure 1: *Optimal K clusters with centers(shown in black) O1,O2, ... OK. Distance of points to their respective cluster center is* $\leq r$*.From triangular inequality pairwise distance between points inside cluster is* $\leq 2r$*. Distance between k optimal clusters is* $> 2r$*.*

Since we are opening new center only when the distance of coming point is greater than 2r from all the centers opened before. Therefore for these $k+1$ centers each pairwise distance is strictly greater than $2r$.
Let us consider the optimal clustering where we have k cluster centers for the given data. Each point $p_i \subset P$ is assigned one of these centers such that distance of p_i from its corresponding cluster center is at most r. Now consider two points assigned to a cluster, distance between each of these points and the center is at most r, then by triangle inequality distance between these two points is at most $2r$. Hence in optimal clustering distance between any two points within a cluster is at most $2r$. Points for which distance between them is strictly greater than $2r$ will be assigned to different clusters. As we have k optimal clusters, we can have no more than k points with pair wise distance among them strictly greater than 2r. This contradicts with our assumption of opening $k+1$ centers with pairwise distance strictly greater than 2r. Hence we will have at most k centers after running *thresholded algorithm.* □

Theorem 2. *Can find* $(2+\epsilon)$ *approximation solution to K-Center in space* $O(\frac{k}{\epsilon}\log\frac{b}{a})$ *if we know that* $a \leq r \leq b$ *for any* $\epsilon > 0$*.*

Proof. Set $\epsilon' = \epsilon/2$. We will run J multiple copies of algorithm in parallel for

$$r = a$$

$$r = a(1+\epsilon')$$

$$r = a(1+\epsilon')^2$$

.

.

.

$$r = a(1+\epsilon')^J$$

Here we choose J such that

$$\begin{aligned} a(1+\epsilon')^J &= b \\ (1+\epsilon')^J &= \frac{b}{a} \\ J &= \log_{(1+\epsilon')} \frac{b}{a} \\ &= \frac{1}{\epsilon'} \log(\frac{b}{a}) \\ &= \frac{2}{\epsilon} \log(\frac{b}{a}), \because \epsilon = 2\epsilon' \end{aligned}$$

Space complexity will be $O(Jk) = O(\frac{k}{\epsilon} \log(\frac{b}{a}))$ if we take all copies storing k centers into account. We drop estimates for which algorithm opens more than k centers and return minimum value for which algorithm opens at most k centers.
Suppose $R_1 = a(1+\epsilon')^i$ is minimum value of r for which algorithm opens at most k centers. Upper bound on error when compared with optimal r is as follow

$$Error \leq 2R_1$$

Also the algorithm for all values less than R_1 must have failed. Hence previous copy i.e. $\frac{R_1}{1+\epsilon'}$ should have failed too. Let OPT be the optimal value of r giving at most K centers. Then we have,

$$\begin{aligned} OPT &> \frac{R_1}{1+\epsilon'} \\ R_1 &< (1+\epsilon')OPT \end{aligned}$$

Now using our previous bound on *Error* we have

$$\begin{aligned} Error &\leq 2R_1 \\ &\leq 2(1+\epsilon')OPT \\ &\leq (2+2\epsilon')OPT \\ &= (2+\epsilon)OPT \end{aligned}$$

□

Theorem 3. *Can find* $(2+\epsilon)$ *approximation solution to K-Center in space* $O(\frac{k}{\epsilon} \log \frac{1}{\epsilon})$ *if we don't know* r*.*

Proof. In this case algorithm is run as follow

1. Read first k items as input and use them as centers. Keep reading input until error is 0 i.e. till points equal to these k centers continue to repeat in stream.

2. As soon as we see a point different from initial k centers we get a non-zero error. We use this as lower bound a for r.

3. Initialize and run in parallel $O(\frac{1}{\epsilon})$ copies of thresholded algorithm for

$$I_0 = a$$

$$I_1 = a(1+\epsilon')$$
$$I_2 = a(1+\epsilon')^2$$

.

.

.

$$I_J = a(1+\epsilon')^J$$

where $(1+\epsilon')^J = O(\frac{1}{\epsilon'})$ and hence $J = O(\frac{1}{\epsilon'}\log\frac{1}{\epsilon})$.

4. If any copy of the thresholded algorithm fails i.e. try to open $k+1$ centers, then we terminate it and start algorithm with an incremented estimate of r. Suppose algorithm fails for some I_i, where $i \in [1, J]$, then we terminate the algorithm for all $I_{i'}$ where $i' \leq i$. We start running a thresholded algorithm for estimates $I_{i'}(1+\epsilon')^{J+1}$ where $i' \in [0, i]$. For these new copies we use previous learned k centers of $I_{i'}$ as initial input.

5. Repeat the above step until the end of stream. On end report the centers for lowest estimate for which algorithm is still running.

As an example of step 4 above, suppose on reading p_{j+1} element of stream $[p_1, p_2, ..., p_j, p_{j+1}, ...p_n]$ I_2 terminates or decides to open $k+1$ centers. Then we will terminate I_0 and I_1 too and start $I'_0 = a(1+\epsilon')^{(J+1)}$, $I'_1 = a(1+\epsilon')^{(J+2)}$ and $I'_2 = a(1+\epsilon')^{(J+3)}$ with already chosen k centers for terminated I_0, I_1, I_2 as inputs respectively. So for I'_2 input now will be $[q_1, q_2, ..., q_k, p_{j+1}, p_{j+2}, ..., p_n]$ where $q_1, q_2, ..., q_k$ were k centers for terminated I_2.
Why does this still gives a good bound?
Suppose after reading new input $[q_1, q_2, ..., q_k, p_{j+1}]$, p_{j+1} is used as a new center and q_2 is assigned to it. One of our original points say p_2 was assigned to q_2 previously. Therefore, now, we can think p_2 is assigned to p_{j+1}. If $l_{2'}$ and l_2 are our new and old estimates respectively. We have

$$\begin{aligned} d(p_2, p_{j+1}) &= d(p_2, q_2) + d(q_2, p_{j+1}) \\ &\leq 2l_2 + 2l_{2'} \\ &\leq 2(l_2 + l_{2'}) \\ &\simeq 2l_{2'}, l_2 << l_{2'} \end{aligned}$$

From above we can see that errors for actual points which we are not storing are negligible under new estimates.

Sketch Analysis

- Suppose end threshold is R and it is updated i times:

$$R_0, R_1 = R_0(1+\epsilon')^{J+1}, R_2 = R_0(1+\epsilon')^{2(J+1)}, ..., R = R_i = R_0(1+\epsilon')^{i(J+1)}$$

- $i = 0, R_0$ gave the best estimate for r. $Q_1 = P_1 = [p_1, p_2, .., p_j]$ here centers in from Q_1 which is subset of P_1 works for the whole input. Following the similar arguments as we did for algorithm when bounds were known. We have

$$Error(Q_1) = Error(P_1) \leq 2R_0$$

$$OPT(Q_1) > \frac{R_0}{(1+\epsilon')}$$

$$Error(Q_1) \leq 2R_0 \leq (2+2\epsilon')OPT(Q_1) \simeq (2+\epsilon)OPT(Q_1)$$

- $i = 1, R_1$ gave the best estimate for r. $Q_2 = [q_1, q_2, ..., q_k, p_{j+1}, p_{j+2}, .., p_{j'}]$, $P_2 = [p_{j+1}, p_{j+2}, .., p_{j'}]$. Terminates with $R_1 = R_0(1+\epsilon')^{J+1}$ but not with $\frac{R_1}{(1+\epsilon')}$. Here centers will be chosen from processing previous centers and some portion of incoming stream.

$$Error(Q_2) \leq 2R_1$$

$$OPT(Q_2) > \frac{R_1}{1+\epsilon'}$$

$$Error(Q_2) \leq 2R_1 = (2+2\epsilon')OPT(Q_2) \simeq (2+\epsilon)OPT(Q_2)$$

- Relationships between $Error(Q_2)$ and $Error(P_1 \bigodot P_2)$ and in between $OPT(Q_2)$ and $OPT(P_1 \bigodot P_2)$

  1 $Error(P_1 \bigodot P_2) \leq Error(Q_2) + Error(Q_1) \leq 2R_1 + 2R_0 = 2R_1 \left(1 + \frac{1}{(1+\epsilon')^{J+1}}\right)$

  2 $OPT(P_1 \bigodot P_2) \geq OPT(Q_2) - Error(Q_1) \geq \frac{R_1}{(1+\epsilon')} - 2R_0 = \frac{R_1}{(1+\epsilon')}\left(1 - \frac{2}{(1+\epsilon')^J}\right)$

  We therefore see $Error(P_1 \bigodot P_2)$ is close to $2+2\epsilon'$ times $OPT(P_1 \bigodot P_2)$.

  For any $i > 1$, we can similarly do the proof via induction.

□

Space Complexity: Since we always maintain $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ copies of K-Center algorithm running for different estimates of optimal distance, the space complexity is $O(\frac{k}{\epsilon}\log\frac{1}{\epsilon}$.

3 K-Median

Points from set $P = \{p_1, p_2, ..., p_n\}$ are coming in streaming fashion. We will find a set of k points $Q \subset P$, $|Q| = k$, that minimizes

$$\sum_{i=1}^{n} d(p_i, c(i))$$

where d is any distance metric and $c(i)$ returns the center ($\subset Q$) of cluster to which p_i is assigned. Above objective function minimizes the total sum of distance of points to nearest cluster. Unlike K-center where only a single point was contributing to error here all the points contribute to error. As all the points contribute to error we also need to keep a count of number of points assigned to a cluster.

3.1 Randomized algorithm

If we know the optimal r and k is given. Set

$$f = \frac{r}{k(1 + log(n))}$$

When considering point x, let δ be the distance to the nearest open center. Open a center at x with probability $\frac{\delta}{f}$, else assign it to the nearest open center. We can also weigh this probability by the number of points present in cluster and open a center with probability $\frac{w\delta}{f}$ where w is proportional to number of points present in nearest cluster.

If we don't know optimal r but k is given, then algorithm is as follow:

1. Read first k distinct points from stream and use them as initial k centers.
2. We will get an error as soon as a distinct $k+1$th point different from k centers is encountered. We will use this error as our initial estimate of r. From this point of the input we will start $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ copies of randomized algorithm with estimates of r similar to the case of K-center algorithm when we didn't know r. As this is a randomized algorithm we will actually start $O(\frac{1}{\epsilon}\log n)$ copies to boost confidence of algorithm.
3. We will declare an individual estimate wrong if error becomes more than $4(1+\epsilon)r$ or tries to open more than $k' \simeq k\frac{\log n}{\epsilon'}$ centers. Similar to K-center we will terminate this copy and start another with an incremented estimate using old weighted k' centers as initial input (initial summary).
4. For final output we will select weighted k' centers from the lowest estimate for which algorithm is still running. We will run K-median offline algorithm on these selected k' centers and will report the learned k centers.

4 K-Means++

K-Means algorithm is applied on given data P to find k centers, Q by minimizing:

$$\sum_{p\in P} \min_{q\in Q} d(p,q)^2$$

Above minimizes the sum of square of within cluster distance for all points. It's different from above algorithms as this can't be applied easily on streaming data, unless we store all the points. We call it *Means* as the center of a cluster is the mean of points assigned to it.

In K-Means algorithm if centers are not initialized correctly then it can get stuck in local optimum easily as shown in Figure 2, corresponding global optimum is shown in Figure 3.

An easy fix to this problem is to initialize centers such that they are furthest from each other. In this approach you select first center at random from the given points, then successively select remaining centers furthest from previous centers. But this approach is sensitive to outliers i.e. a single furthest point will be learned as a cluster in itself. As illustrated in Figure 4, a single point located on far right is selected as cluster in itself.

Algorithm 1 K-Means (Lloyd's)

Select k centers at random from within the P
while centers don't converge (clustering don't change) **do**
  Assign each point to nearest of all k centers
  Update each center to mean of points assigned to it.
end while
return converged k centers

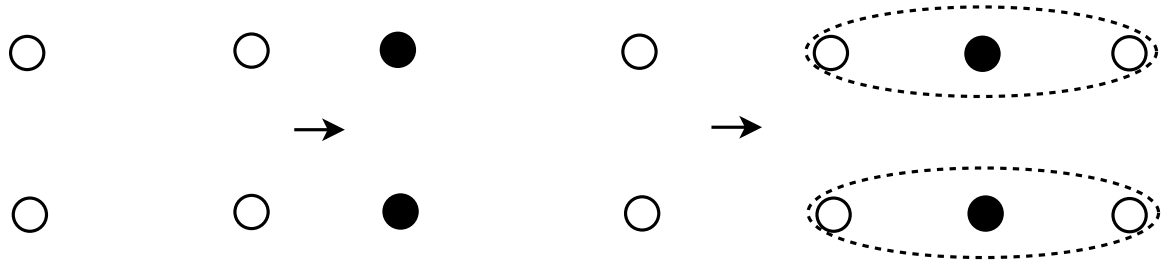

Figure 2: *K-Means coverging to locally optimum clusters.*

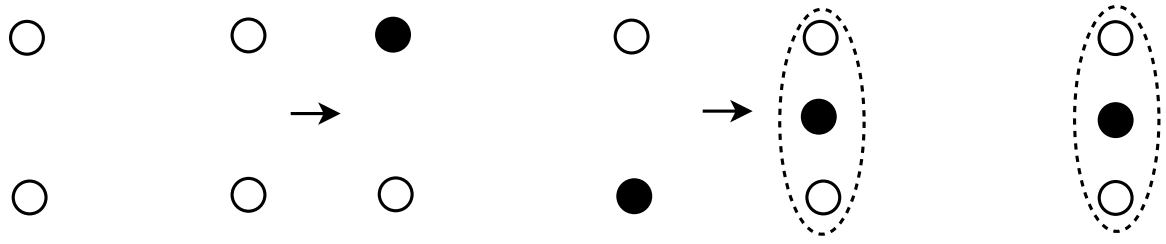

Figure 3: *K-Means coverging to globally optimum clusters.*

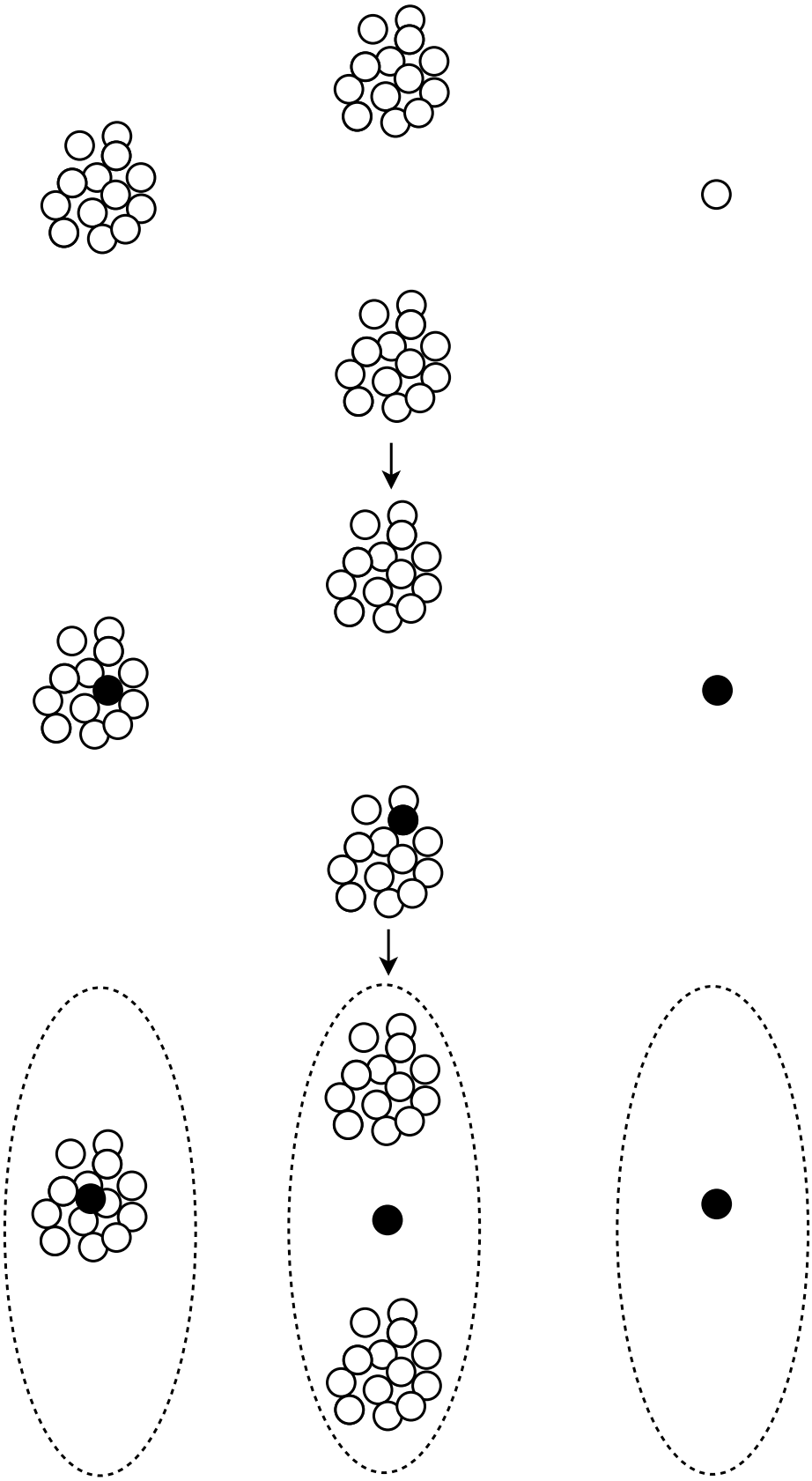

Figure 4: *Furthest point as center approach giving an outlier as cluster in itself.*

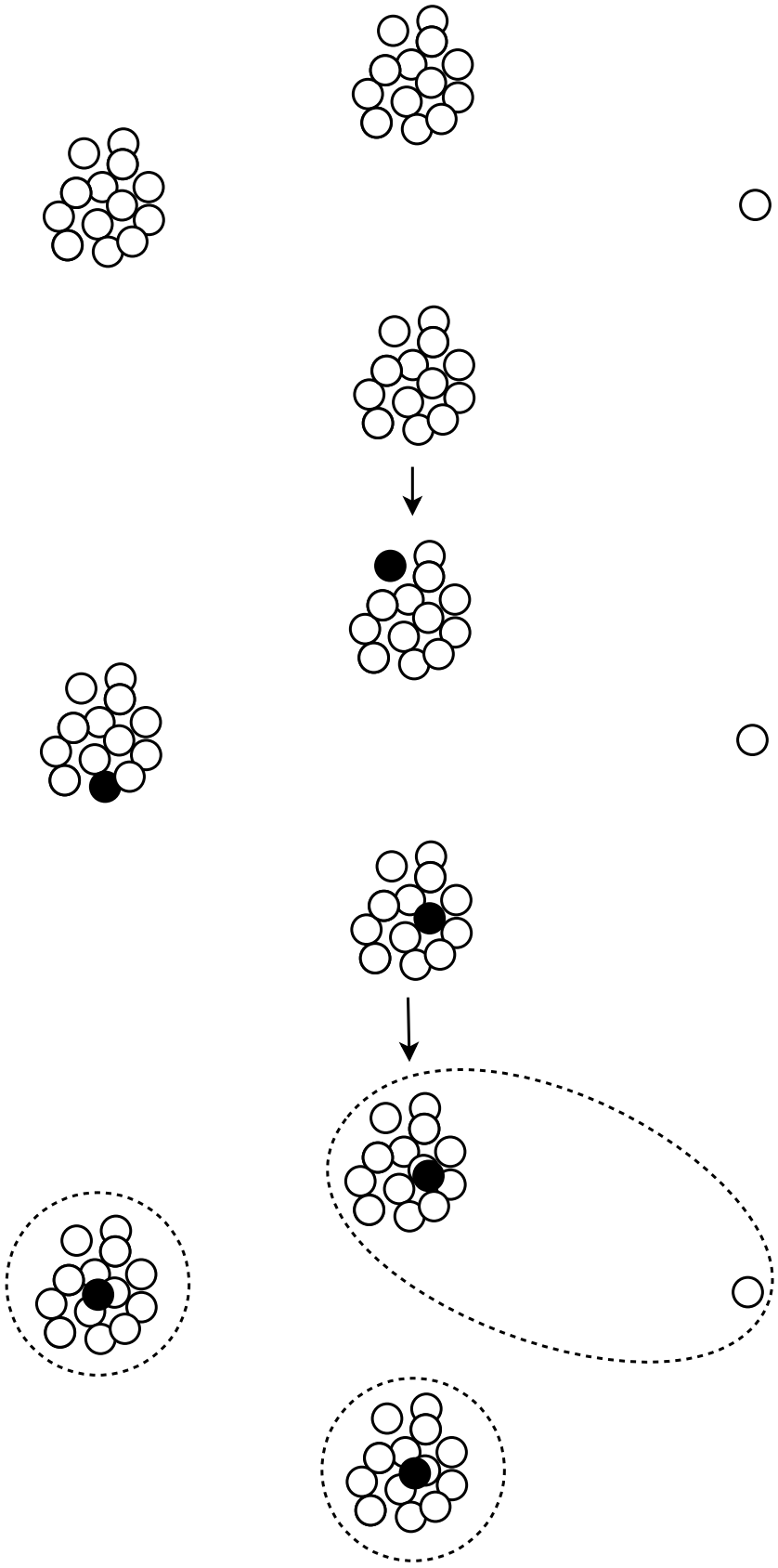

Figure 5: *K-Means++ is not sensitive to outliers. Chances of outlier being selected as a center in step 2 are low, as other points also has some probability to get selected.*

K-Means++ algorithm interpolate between the two methods of random selection and furthest point approach. Let $D(x)$ be the distance between x and the nearest cluster center. Sample points proportionally to $(D(x))^2 = D^2(x)$ for initializing centers. After random selection of first center it selects remaining centers with probability proportional to square of distance between candidate point and nearest cluster center. By inductive proof K-Means++ is shown to be $\Theta(logk)$ approximation in expectation. We can see in Figure 5 that chances of the furthest point on right being selected as center are reduced as other points now has some probability to be picked up as center. In general the number of outliers in a data is generally small compared to overall data, hence probability of selection of outliers as center will be lower compared to that of remaining points.

References

[1] Tight Results for Clustering and Summarizing Data Streams, Sudipto Guha

[2] Better streaming algorithms for clustering problems, Moses Charikar, Liadan O'Callaghan, Rina Panigrahy

[3] Arthur, D., & Vassilvitskii, S. (2007). K-means++: the advantages of careful seeding. In 2007 ACM-SIAM symposium on discrete algorithms (SODA 07).